Using PETSc Solvers in BOUT++

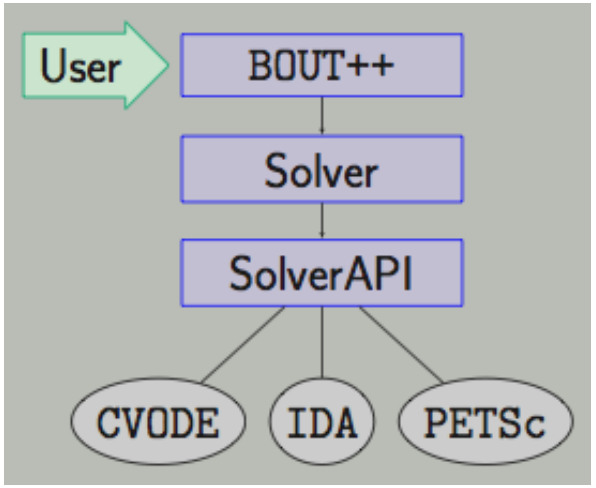
Lois Curfman McInnes, Hong Zhang and the PETSc Team

Mathematics and Computer Science Division
Argonne National Laboratory

BOUT++ Workshop
Livermore, CA
September 14-16, 2011

BOUT++ Application:

using SUNDIALS (LLNL) and PETSc (ANL)



Current emphasis:

- Implicit time integration
- Newton-Krylov nonlinear solves
- Jacobian–free variant
- Preconditioner options:
  - algebraic approaches using sparse finite difference Jacobian evaluation via coloring
  - or user-provided preconditioners

Implementation:

- solver_type=cvode/ida/petsc
- PETSc variant has several options:
  - PETSc calls PETSc/TS or
    SUNDIALS/CVODE
    - SUNDIALS/CVODE can call PETSc/PC

See upcoming presentations:

C. Woodward (SUNDIALS) and P. Narayan (performance analysis)

Newton's Method

Newton
nonlinear solver

Based on multivariate Taylor expansion:

$$F(u^{l+1}) = F(u^l) + F'(u^l)(u^{l+1} - u^l)$$
$$+ \text{higher order terms}$$

$$F'(u^{l-1})\,\delta u^l = -F(u^{l-1})$$
$$u^l = u^{l-1} + \lambda\,\delta u^l$$

- Can achieve quadratic convergence when sufficiently close to solution
- Can extend radius of convergence with line search, trust region, or continuation methods (e.g., pseudo-transient continuation, mesh sequencing)

Krylov Methods

Krylov accelerator

- Projection methods for solving linear systems, $Ax=b$, using the Krylov subspace

$$K_j = span(r_0, Ar_0, Ar_0^2, ..., Ar_0^{j-1})$$

- Require A only in the form of matrix-vector products
- Popular methods include CG, GMRES, TFQMR, BiCGStab, etc.
- In practice, preconditioning typically needed for good performance

Challenges in Preconditioning

- Cluster eigenvalues of the iteration matrix (and thus speed convergence of Krylov methods) by transforming $Ax=b$ into an equivalent form:

$$B^{-1}Ax = B^{-1}b \text{ or } \qquad (AB)^{-1}(Bx) = b$$

where the inverse action of B approximates that of A, but at a smaller cost

- How to choose B so that we achieve efficiency and scalability? Common strategies include:
  - Lagging the evaluation of B
  - Lower order and/or sparse approximations of B
  - Parallel techniques exploiting memory hierarchy, e.g., additive Schwarz
  - Multi-level methods
  - User-defined custom physics-based approaches

The Need for Derivatives

$$F'(u^{l-1})\,\delta u^{l} = -F(u^{l-1})$$ ⟸ Solve approximately using a preconditioned Krylov method

- Newton-Krylov methods require derivatives in the form of Jacobian-vector products, $F'(u)v$
- Also typically require $F'(u)$ (or a "cheaper" approximation) for use in preconditioning
- Options: Can provide either $F'(u)$ or $F'(u)v$ via
  - Analytic code (written by application developer)
  - Sparse finite difference approximation (FD)
  - Automatic differentiation (AD), see www.autodiff.org

} Can be provided by libraries

Matrix-free Jacobian-Vector Products

- **Approaches**
  - Finite differences (FD)
    - $F'(x)\, v = [\, F(x+hv) - F(x)] / h$
    - costs approximately 1 function evaluation
    - challenges in computing the differencing parameter, *h;* must balance truncation and round-off errors
  - Automatic differentiation (AD)
    - costs approx 2 function evaluations, no difficulties in parameter estimation
    - e.g., ADIFOR & ADIC
- **Advantages**
  - Newton-like convergence without the cost of computing and storing the true Jacobian
  - In practice, still typically perform preconditioning
- **Reference**
  - D.A. Knoll and D.E. Keyes, Jacobian-free Newton-Krylov Methods: A Survey of Approaches and Applications, 2004, *J. Comp. Phys*., 193: 357-397

Example: ~BOUT/examples/drift-instability

on a MacAir with 2.13 GHz Inel core 2 Duo and 2GB MHz DDR3 memory

mpiexec -n 4 ./2fluid solver_type=petsc -ts_type sundials -pc_type <pc> [options]

Preconditioner	NONE	LU (MUMPS)	ASM+ LU/ILU	BJACOBI +ILU
Internal time steps	**1278**	**71**	**82/101**	**89**
Calls to rhs function	2409	94	102/137	108
Linear solver setup	0	17	18/21	14
Error test failures	3	5	5/4	3
Nonlinear iterations	**2408**	**93**	**101/136**	**107**
Nonlinear convergence failure	554	0	0/3	1
Linear iterations	**11591**	**77**	**165/348**	**298**
Linear convergence failures	2061	0	0/23	3
Preconditioner evaluations	0	17	18/21	14
Preconditioner solves	0	153	257/475	398
Finite diff. Jacobian-vec product	11591	77	165/348	298
Run time	**4m 9s**	**2m 37s**	**2m 11s / 2m 19s**	**1m 30s**

PETSc –

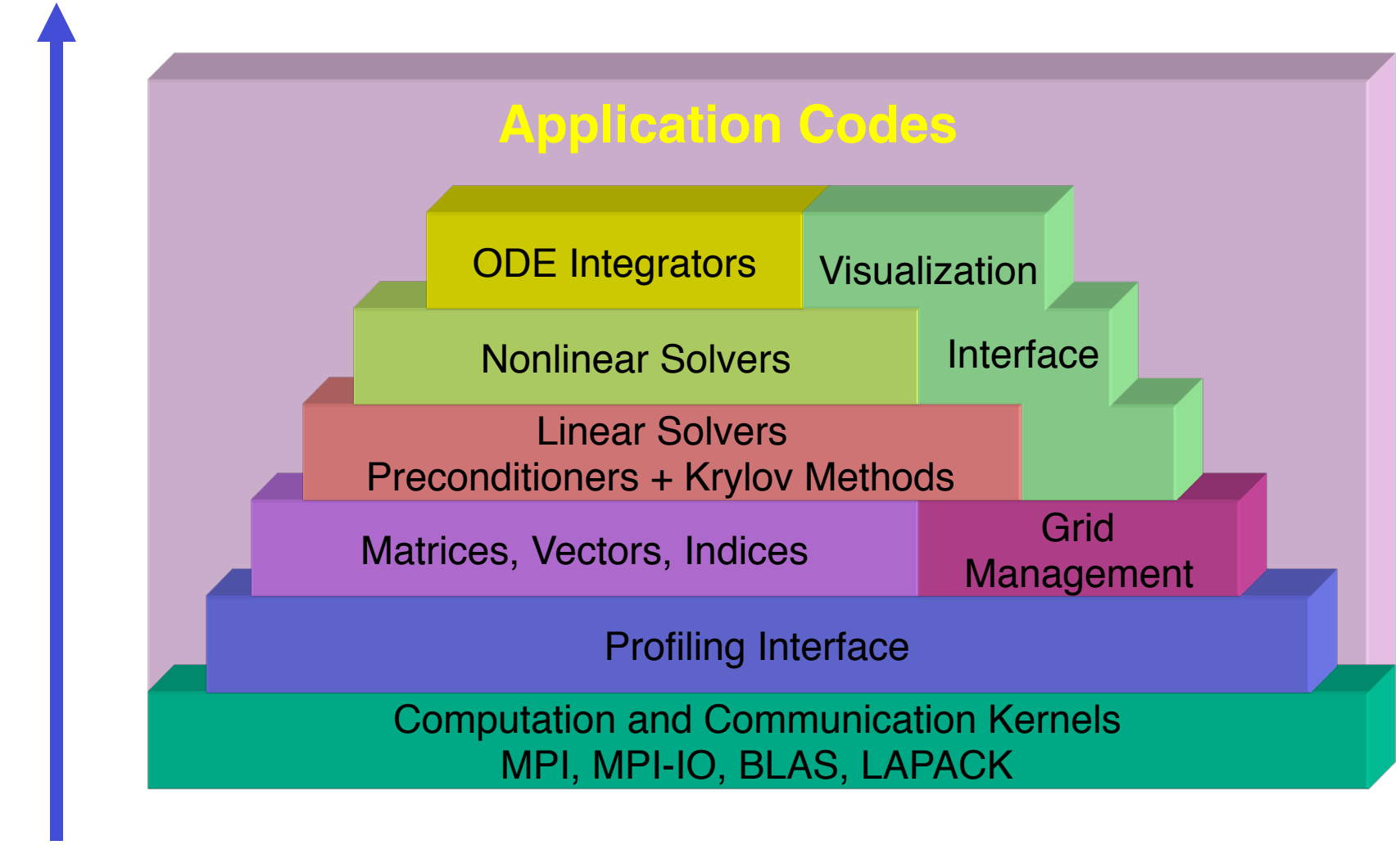
Portable Extensible Toolkit for Scientific computation

- High-performance software for the scalable (parallel) solution of scientific applications
- PETSc History
  - Begun in 1991
  - Over 20,000 downloads since 1995 (version 2)
  - Currently about 400 downloads per month
- PETSc Funding and Support
  - primarily Department of Energy
  - National Science Foundation

Portable Extensible Toolkit for Scientific computation

- Portable to any parallel system supporting MPI
  - Tightly coupled systems
    - Cray XT5, SGI Origin, IBM SP, HP 9000,…
  - Loosely coupled systems
    - PCs running Linux or Windows
- Free for everyone, including industrial users.
- Download from www.mcs.anl.gov/petsc
- Extensive documentation
- Hundreds of tutorial-style examples
- Support via email: petsc-maint@mcs.anl.gov
- Usable from Fortran90, C, C++, Python

Portable Extensible Toolkit for Scientific computation



Portable Extensible Toolkit for Scientific computation

- Parallel vector/array operations
- Numerous (parallel) sparse matrix formats
- Numerous linear solvers
- Nonlinear solvers
- ODE integrators
- Parallel grid/data management
- Provides common interface for
  SUNDIALS/CVODE, hypre, SuperLU, MUMPS,…
- **Allows switching of virtually all solvers at runtime**

Portable Extensible Toolkit for Scientific computation

Interfaced Packages:

- LU, Cholesky, ILU, ICC
  SuperLU/SuperLU_DIST, MATLAB, PLAPACK, UMFPACK, MUMPS
- Algebraic multigrid
  BoomerAMG (part of hypre), ML (part of Trilinos)
- Parititioning: Parmetis, Chaco, Jostle, Party, Scotch
- ODE integrators: Sundials/CVODE
- Eigenvalue solvers: BLOPEX
- FFTW
- SPRN
- …

Child Packages of PETSc: Have PETSc's style of programming

- SLEPc: scalable eigenvalue/eigenvector solver packages
- TAO: scalable numerical optimization algorithms

Portable Extensible Toolkit for Scientific computation

Solvers, (parallel) debugging aids, low-overhead profiling

- It is not possible to select the most effective solver a priori
  - What will deliver best/competitive performance for a given physics, problem size, discretization, and architecture?
- PETSc was developed as a platform for experimentation
  - models, discretization, algorithms, solvers
  - algebra of composition so new solvers can be created at runtime
- Important to keep solvers decoupled from physics, discretization and processor partitioning because we also experiment with those

Portable Extensible Toolkit for Scientific computation

Who Uses PETSc?

- Computational Scientists
  - PyLith (TECTON), Underworld, Columbia group, PFLOTRAN, etc.
- Algorithm Developers
  - Iterative methods and preconditioning researchers
- Package Developers
  - SLEPc, TAO, MagPar, StGermain, Dealll, PETSc-FEM

Portable Extensible Toolkit for Scientific computation

Applications of PETSc: (see more at www.mcs.anl.gov)

- Nano-simulations
- Biology/Medical
- Cardiology
- Imaging and Surgery
- Fusion
- Geosciences
- Environmental/Subsurface Flow
- Computational Fluid Dynamics
- Wave propagation and the Helmholz equation
- Optimization
- Software engineering
- Algorithm analysis and design

What Can We Handle?

- PETSc has run implicit problems with over 500 billion unknowns
  - PFLOTRAN (flow in porous media)
  - UNIC (nuclear energy) on BG/P and XT5

- PETSc has run on over 290,000 cores efficiently
  - UNIC on IBM BG/P Intrepid at ANL
  - PFLOTRAN on the Cray XT5 Jaguar at ORNL

- PETSc applications have run at 22 Teraflops
  - PFLOTRAN
  - UNIC

- PETSc also runs on your laptop

Outline

- Overview of PETSc user interface:
  - Linear solvers: KSP
  - Nonlinear solvers: SNES
  - ODE solvers: TS
  - Profiling and debugging

- What's New in PETSc

The PETSc Programming Model

- Distributed memory, “shared-nothing”
  - Requires only a standard compiler
  - Access to data on remote machines through MPI
- Hide within objects the details of the communication
- User orchestrates communication at a higher abstract level than direct MPI calls

Getting Started

```
PetscInitialize();
ObjCreate(MPI_comm,&obj);
ObjSetType(obj, );
ObjSetFromOptions(obj, );

ObjSolve(obj, );
ObjGetxxx(obj, );

ObjDestroy(obj);
PetscFinalize();
```

PETSc Numerical Components

Nonlinear Solvers (SNES)		
Newton-based Methods		Others
Line Search	Trust Region	

Time Steppers (TS)			
Euler, RK, SSP	Theta, Alpha, GL	Pseudo Time Stepping	Others

Krylov Subspace Methods (KSP)							
GMRES	CG	CGS	Bi-CG-STAB	TFQMR	Richardson	Chebychev	Others

Preconditioners (PC)						
Additive Schwartz	Block Jacobi	Shell	ILU, LU	MG	Field Split	Others

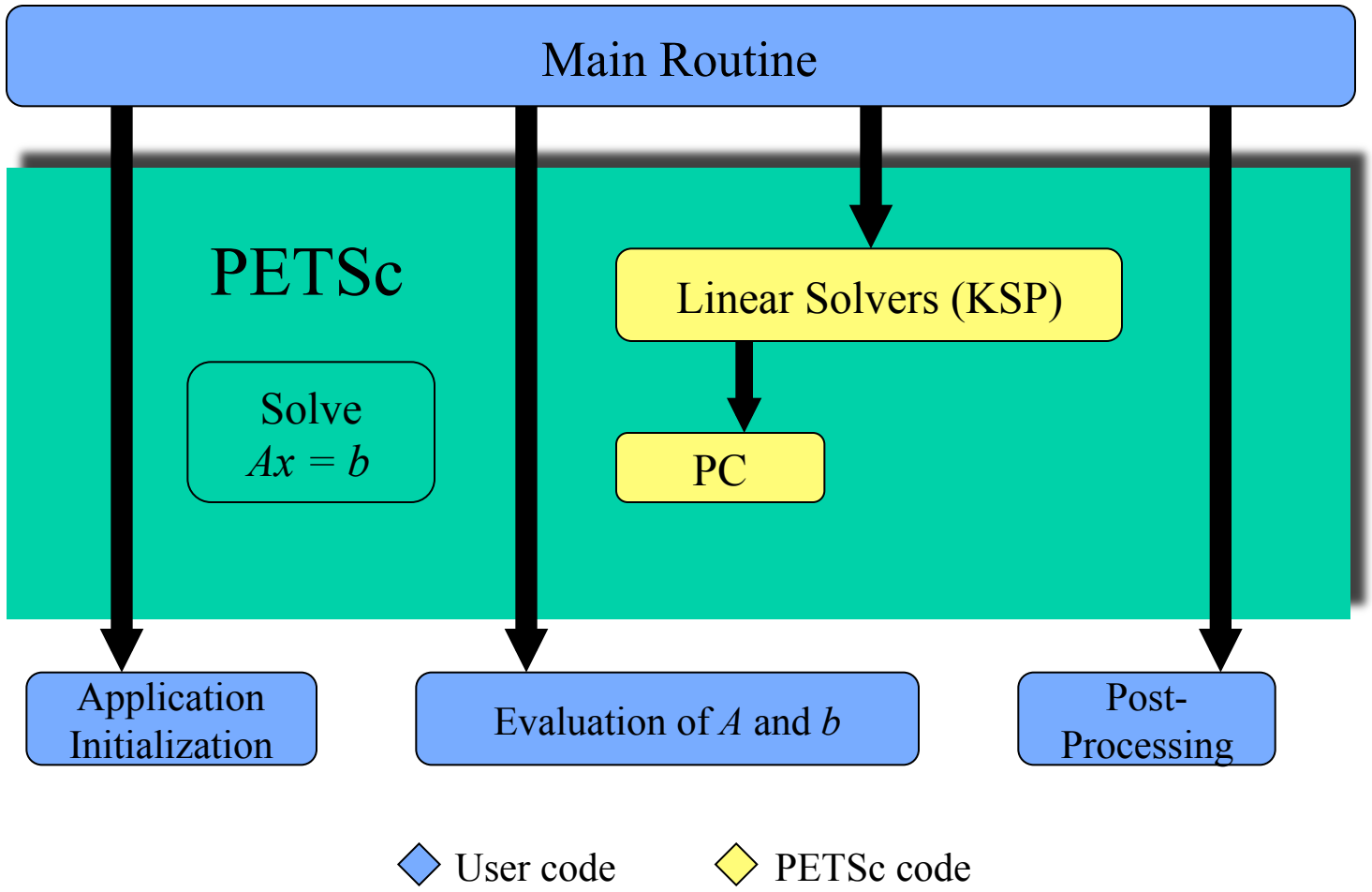
Matrices (Mat)					
Compressed Sparse Row (AIJ)	Blocked Compressed Sparse Row (BAIJ)	Block Diagonal (BDIAG)	Dense	Matrix-free	Others

Distributed Arrays (DA)

Vectors (Vec)

Index Sets (IS)			
Indices	Block Indices	Stride	Others

Linear Solver Interface: KSP



Setting Solver Options at Runtime

- -ksp_type [cg,gmres,bcgs,tfqmr,…]
- -pc_type [lu,ilu,jacobi,sor,asm,…]

- -ksp_max_it <max_iters>
- -ksp_gmres_restart <restart>
- -pc_asm_overlap <overlap>
- -pc_asm_type [basic,restrict,interpolate,none]
- etc ...

Recursion: Specifying Solvers for Schwarz Preconditioner Blocks

- Specify KSP solvers and options with "-sub" prefix, e.g.,
  - Full or incomplete factorization

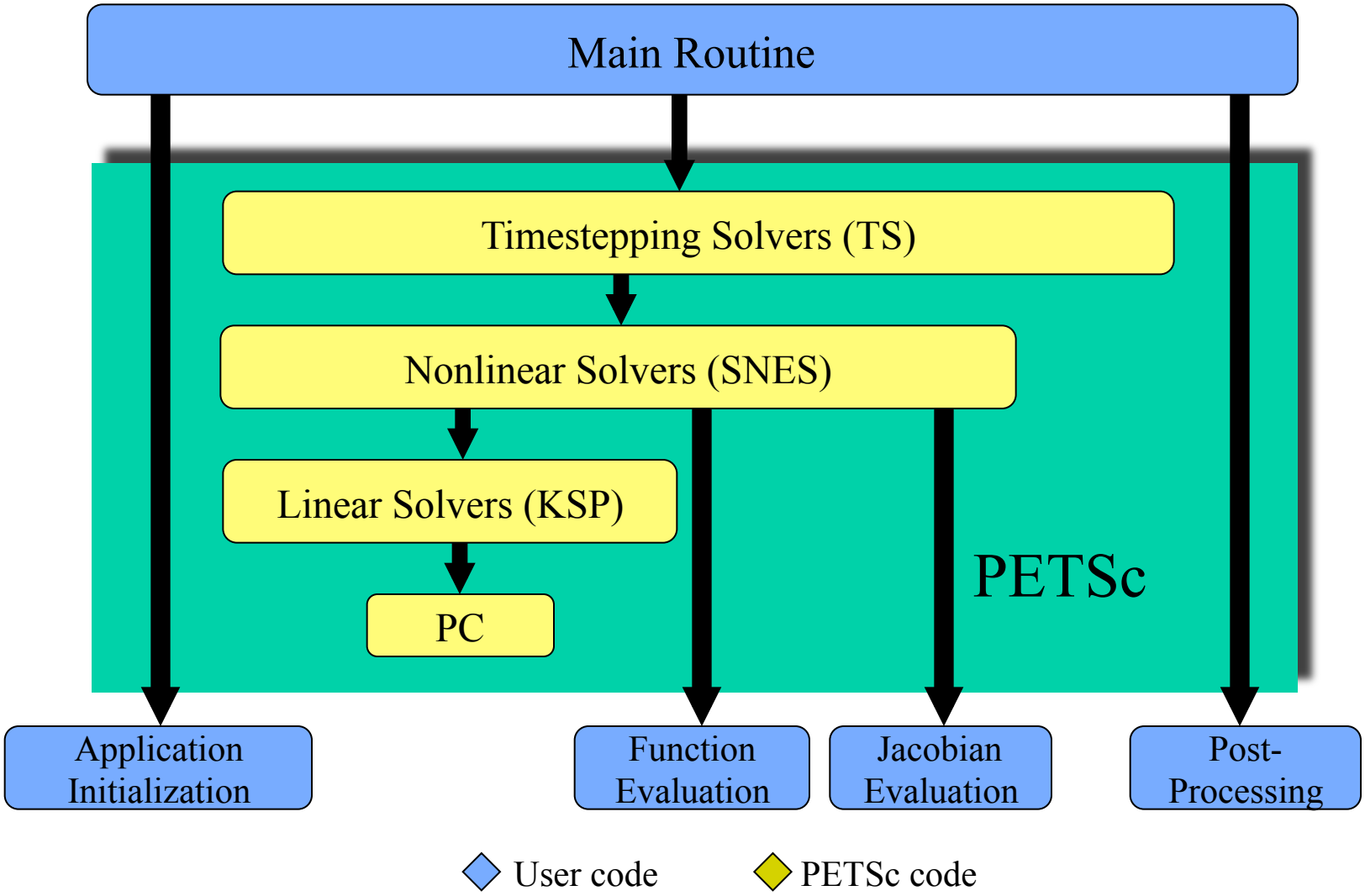
    -sub_pc_type lu

    -sub_pc_type ilu -sub_pc_ilu_levels <levels>
  - Can also use inner Krylov iterations, e.g.,

    -sub_ksp_type gmres -sub_ksp_rtol <rtol>

    -sub_ksp_max_it <maxit>

Flow of Control for PETSc Solvers



Nonlinear Solver Interface: SNES

Goal: For problems arising from PDEs, support the general solution of $F(u) = 0$

User provides:

- Code to evaluate $F(u)$
- Code to evaluate Jacobian of $F(u)$ (optional)
  - or use sparse finite difference approximation
  - or use automatic differentiation
    - AD support: automated interface to ADIFOR and ADIC
    - See www.mcs.anl.gov/autodiff

SNES: Basic Usage

- SNESCreate( ) - Create SNES context
- SNESSetFunction( ) - Set function eval. routine
- SNESSetJacobian( ) - Set Jacobian eval. routine
- SNESSetFromOptions( ) - Set runtime solver options for [SNES, KSP,PC]
- SNESSolve( ) - Run nonlinear solver
- SNESView( ) - View solver options actually used at runtime (alternative: -snes_view)
- SNESDestroy( ) - Destroy solver

ODE Solver Interface: TS

ODE: $du/dt = F(u, t)$ or

DAE: $G(u, du/dt, t) = F(u, t)$

User provides:

- Code to evaluate function
- Code to evaluate Jacobian (optional)
  - or use sparse finite difference approximation

TS: Basic Usage

- TSCreate( )
- TSSetProblemType()
- TSSetType()
- TSSetRHSFunction( ) and/or TSSetIFunction()
- TSSetRHSJacobian( ) or TSSetIJacobian()
- TSSetInitialTimeStep()
- TSSetDuration()
- TSSetFromOptions( )

- TSSolve( ) - Time stepping
- TSView()
- TSDestroy( )

Uniform Access to All PETSc Solvers

- -ksp_type [cg, gmres, bcgs, tfqmr,…]
- -pc_type [lu, ilu, jacobi, sor, asm,…]
- -ksp_monitor_true_residual
- -ksp_converged_reason

- -snes_type [ls,…]
- -sles_ls <parameters>
- -snes_rtol <> -snes_atol <> -snes_stol <>
- -snes_monitor
- -snes_converged_reason

- -ts_type [sundials, theta, beuler…]
- -ts_theta_[option]
- -ts_dt <> -ts_max_time <>
- -ts_view

Uniform Access to External Solvers

mpiexec –n <np> ./petsc_program [petsc options]

- -ts_type sundials
- -ts_sundials_type [adams, bdf]
- -ts_sundials_linear_tolerance <tolerance>
- -ts_sundials_monitor_steps
- -pc_type lu
- -pc_factor_mat_solver_package superlu_dist
- -mat_superlu_dist_[option]

mpiexec –n <np> ./slepc_program [slepc/petsc options]

- -eps_type [power, arnoldi, lapack, blopex,…]
- -eps_nev <> -eps_ncv <> -eps_ smallest_magnitute
- -st_type [shift, sinvert,…] –st_shift <> -st_ksp_type <>

Reference: http://www.grycap.upv.es/slepc

PETSc Programming Aids

- Correctness Debugging
  - Automatic generation of tracebacks
  - Detecting memory corruption and leaks
  - Optional user-defined error handlers
- Performance Profiling
  - Integrated profiling using -log_summary
  - Profiling by stages of an application
  - User-defined events

What's New in PETSc?

- User-friendly APIs
  - Python Bindings: petsc4py (Dalcin at CIMEC), Elefant (SML group at NICTA)
  - PETSc-MATLAB
- Architecture–aware numerical algorithms and implementations
  - Optimization for GPUs
    http://www.mcs.anl.gov/petsc/petsc-as/features/gpus.html
  - Hybrid MPI + pthreads
    http://www.mcs.anl.gov/petsc/petsc-as/features/threads.html
  - New algorithms and implementations that address memory scalability, efficient data accessing, e.g., iBiCGStab
- Variational inequality solvers
  - SNESSetType(VI), SNESVISetVariableBounds(), …
- Mathematical solvers for multiphysics
  - PCFieldSplit: support for physics-based preconditioning
- Time integration, DAE (operator spliting G(u, du/dt, t) = F(u, t))

PETSc Solvers for Multiphysics:

Exploiting physics knowledge in custom preconditioners …

New PCFieldSplit simplifies multi-model algebraic system specification and solution.

UEDGE runtime option:

```
prec['7']=['-pc_type fieldsplit','-pc_fieldsplit_block_size 5',
'-pc_fieldsplit_type additive','-pc_fieldsplit_0_fields 0,1,2,3
','-pc_fieldsplit_1_fields 4','-fieldsplit_0_pc_type asm','-fie
ldsplit_0_sub_pc_type lu','-fieldsplit_0_sub_pc_factor_mat_solv
er_package mumps','-fieldsplit_1_pc_type lu','-fieldsplit_1_pc_
factor_mat_solver_package mumps']
```

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

base ordering (all variables per mesh point)

physics-based reordering (for PC only)

$$\begin{pmatrix} A_1 & A_2 \\ A_3 & A_4 \end{pmatrix}$$

components are separated

A_1 Non-neutral terms only

A_2 Neutral to Non-neutral

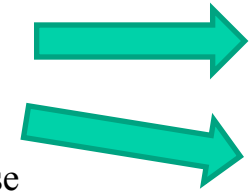
A_3 Non-neutral to neutral

A_4 Neutral terms only

Leveraging knowledge of the different component physics in the system produces a better preconditioner.

$$\begin{pmatrix} A_1^{-1} & \\ & A_4^{-1} \end{pmatrix}$$

additive componentwise preconditioner



A_1^{-1} Additive Schwarz, LU on blocks

A_4^{-1} Full LU solve via MUMPS

Additive Schwarz provides scalability

LU handles the neutrals only

Physics-based Preconditioning in UEDGE

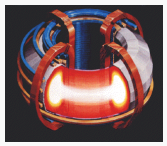
Scalable Preconditioners for Coupled Plasma/Neutral Boundary Transport Simulations

Michael McCourt[1,2], Tom Rognlien[3], Lois Curfman McInnes[2], Hong Zhang[2]

[1]Cornell University, [2]Argonne National Laboratory, [3]Lawrence Livermore National Laboratory

Office of Science
U.S. Department of Energy

Introduction

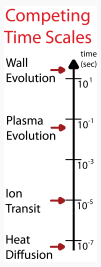


- We study simulations of the edge region of a Tokamak magnetic confinement fusion reactor using UEDGE.
  - UEDGE is a 2D parallel edge plasma application developed by T. Rognlien et al. (LLNL)
- UEDGE is one of the edge plasma transport components in FACETS.
  - FACETS: Framework Application for Core-Edge Transport Simulations based at Tech-X Corporation
  - PI: John Cary, `https://www.facetsproject.org`
  - FACETS goal: Strong coupling between core, edge and wall Tokamak regions during simulation

Governing Physics

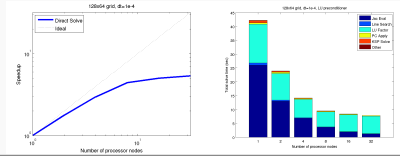
UEDGE uses a fluid transport model, conserving particles, momentum and energy.

- Simulations use $\Delta t \in [10^{-4}, 10^{-3}]$ sec, appropriate for coupling to time-dependant core models.
  - Coupled plasma/neutral simulations involve a large range of spatial and temporal scales.
- Several coupled variables interact in the basic simulation:
  - Deuterium ion D^+ temperature
  - Deuterium ion D^+ density
  - Deuterium ion D^+ parallel velocity
  - Electron e temperature
  - Neutral Deuterium D density
- Strong nonlinearities can yield ill-conditioned simulations
- Impurities in the plasma arise from:
  - Plasma sputtering of material walls, and
  - Edge transport competing with ionization/recombination.
  - Solving each charge state (or bundle) creates large systems.



Algorithms

- Implicit time discretization with nonlinear solves via preconditioned Jacobian-free Newton-Krylov
  - The choice of preconditioner is vital to achieving scalability
- PETSc is used to conduct the simulation in parallel
- Early experiments showed limited scalability
  - The direct solver becomes overwhelmed by the cost of LU factorization and associated communication.



Motivating a Physics Preconditioner

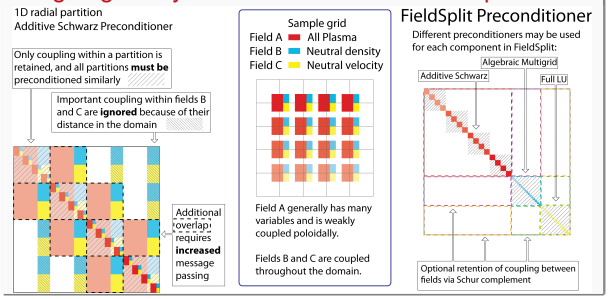
Physics issues to consider for computational stability/accuracy/efficiency:

- Solving plasma and neutral equations on the same mesh simplifies their strong coupling; this is helpful to ensure an accurate simulation.
  - Wall particle recycling and ionization can result in long physical times to reach equilibrium; this competes with the fast edge plasma transport.
- To accommodate the dominant plasma transport, the discretization is highly anisotropic.
  - For standard Δt the plasma terms are well-conditioned enough to use an easily scalable preconditioner such as Additive Schwarz.
- However, neutral collisional diffusive transport is isotropic, and very ill-conditioned on an anisotropic mesh.

Radial width is much greater than poloidal

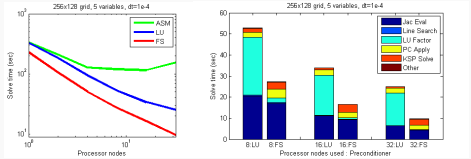
This physical knowledge implies that *separate* methods should be used to precondition the plasma and neutral terms within the nonlinear solver.

Designing a Physics Preconditioner: FieldSplit



Results: FieldSplit Preconditioning

- Initial FieldSplit structure - 2 separate fields preconditioned individually:
  - Field 1: 4 plasma terms solved with Additive Schwarz
  - Field 2: 1 neutral term solved with Algebraic Multigrid
- Component preconditioners are added together
- Coupling terms between fields are disregarded during preconditioning.

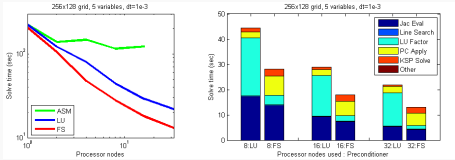


Solver	NP=32 results KSP its	Time (sec)
ASM	1585	154
LU	7	25
FS	44	10

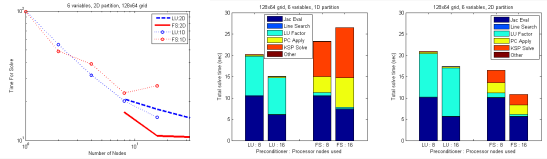
- By handling the troublesome fields (neutral gases) separately we can use a more scalable solver on the easier fields (plasma).
- 1D partitioning allows for the majority of fields (plasma) to be on their more optimal domain.

Results: Scalability for More Complex Problems

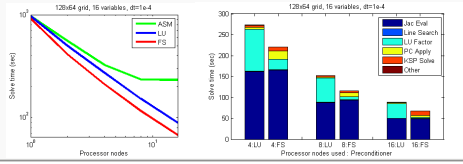
- FieldSplit performs well for larger time steps, so long as the plasma terms can still be solved scalably.



- Initially the neutral D velocity was computed with a simpler algebraic model. Below are results with its inclusion in the nonlinear solve.
  - A 2D partitioning is preferred for this problem, which is first available at NP=8.



- We also enjoy improved performance in the presence of a Neon impurity and the 11 new individual fields added as a result.



Conclusions

- FieldSplit overcomes a major obstacle to parallel scalability for an implicit coupled neutral/plasma edge model.
  - This allows greatly reduced runtimes when using multiple processors.
  - Little code manipulation is required.
- Jacobian-free Newton-Krylov within PETSc using FieldSplit preconditioning provides flexibility for optimizations such as
  - Redundant preconditioning on comparatively small fields,
  - Variable Additive Schwarz overlap, and
  - Jacobian lagging both within and across time steps.

Future Work

- As different species (e.g., He and C) are added and larger Δt used, how can FieldSplit be optimized?
- The goal of the FACETS project is Core-Edge-Wall coupling
  - How can this physics preconditioning be applied in a multiphysics setting?
  - What techniques developed here can be used in 3D edge codes, e.g., BOUT++?
- Coupling terms can be retained via the Schur complement.
  - Cost is greater than Additive FieldSplit.
  - While not needed so far, will this coupling be useful in multiphysics preconditioning?

Acknowledgements

Special thanks to Charles Van Loan, Satish Balay, and Barry Smith.

Conclusions

PETSc can help you

- Easily construct a code to test your ideas
  - tools to aid code construction, management, debugging
- Scale an existing code to large or distributed machines
- Incorporate more scalable or higher performance algorithms
  - such as domain decomposition or multigrid
- Tune your code to new architectures
  - using profiling tools and specialized implementations

How Can We Help?

- Provide documentation:
  - http://www.mcs.anl.gov/petsc
- Quickly answer questions
- Assist with installation
- Guide large-scale flexible code development
- Answer email at petsc-maint@mcs.anl.gov